Literature survey for Learning to rank

Ruoyao Ding
Computer and Information Science Department
University of Delaware
Newark Delaware 19716
ryding@udel.edu

Abstract

This is a survey on the topic of Learning to rank (LTR). In this survey, three categories of LTR approaches: Pointwise approaches, Pairwise approaches, and Listwise approaches will be introduced. Two classic algorithms in each of these categories will be discussed in detail.

1 Introduction

Ranking is the central problem in the field of information retrieval, the task is to rank all the documents from the available sets for a given query, in accordance with their relevance. It has widespread applications including commercial search engines and recommend system.

Learning to rank is to use Machine Learning methods to train a machine learning model, which can find out relevance between the relevant documents in context of given user query, and place them in order of their relevance. In training, both the queries and documents are provided, each query is associated with a perfect ranking list of documents, a ranking model is then created using the training data, as shown in Figure 1.

Learning to rank has three main categories: Pointwise approaches, Pairwise approaches, and Listwise approaches.

The Pointwise approaches are the earliest approaches that researchers considered. The basic idea of these approaches is to map the documents' ordinal scales into numeric values, and then solve the problem as a standard regression problem. Each document will be given a ranking score, and documents will be ranked based on these scores. In Pointwise approaches, a document-query pair is considered as one training instance.

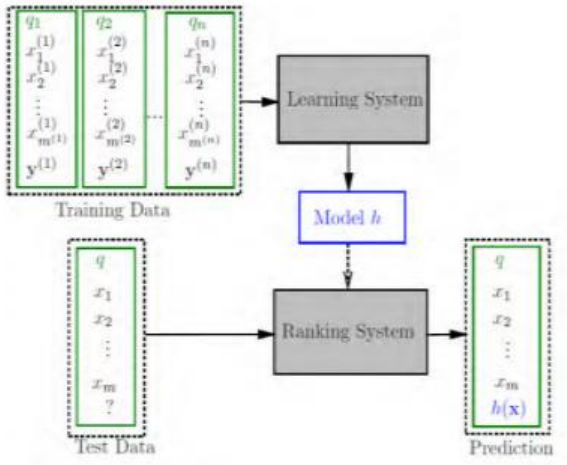

Figure 1: Structure of LTR system

The Pairwise approaches try to compare the relevance of every two documents, then rank all the documents based on all these comparison results. In this way, the ranking problem can be solved as a binary classification problem, by assigning a label to each document in the document-pair, representing their relative relevance. In Pairwise approaches, two ranked documents and one query is considered as one training instance.

The idea of Listwise approaches is similar with the idea of pairwise approaches. It tries to directly compare the relevance of list of documents, instead of trying to get one ranking

score for each document as Pointwise methods do. In Listwise approaches, one query and a list of ranked documents is considered as one training instance.

In section 2, two Pointwise based algorithms, McRank and PRank will be introduced. In section 3, we will discuss two Pairewise based algorithms, RankNet and LamdaRank. Section 4 will focus on two Lisiwise based algorithms, ListNet and BoltzRank. Section 5 will be the conclusion.

2 Pointwise Approaches

In this section, we will introduce two classic pointwise methods, McRank and Pranking.

2.1 McRank

In McRank (P. Li et al., 2007), they cast the ranking problem as a multiple classification problem, by assigning the ranking score of each documents based on the classification result.

This approach is motivated by the fact that perfect classification result in perfect DCG score, and the DCG errors are bounded by classification errors.

DCG (Discounted Cumulative Gain) score is a criterion used in evaluating ranking result, it's computed as follow:

$$\mathrm{DCG} = \sum_{i=1}^{n} c_i(2^{y_{w_i}} - 1) = \sum_{i=1}^{n} c_{\pi_i}(2^{y_i} - 1)$$

Where π_i is the rank order, the smaller π_i is, the higher it will be ranked; y_i is the relevance level, the larger y_i is, the higher relevance the document has. c_i is a non-increasing function of i, it's typically set as:

$$c_i = \frac{1}{\log(i+1)}$$

As we can see, perfect classifications will lead to perfect DCG scores (view y_i as class, if use K-classes classification, let $y_i \in (0, 1, 2, \ldots, K\text{-}1)$), so the ranking problem can be casted as multiple classification problem.

Then they proposed a function to convert classification results into ranking scores:

$$S_i = \sum_{k=0}^{K-1} p_{i,k} T(k)$$

$p_{i,k}$ is the probabilities that document, which is ranked i, has the relevance level k. $T(k)$ is an increasing function of the relevance level k, usually $T(k) = k$, or $T(k) = 2^k$.

A surrogate loss function for multiple classifications is used:

$$\sum_{i=1}^{n} \sum_{k=0}^{K-1} \log(p_{i,k}) 1_{y_i=k}$$

With this loss function, a boosting tree algorithm is used to learn the class probability: $p_{i,k}$, and finally all the documents are ranked based on the value of S_i.

2.2 PRank

PRank (Koby et al., 2002) regards a query and document pair as one instance, each instance is corresponding to a rank level $r = \{1, 2, \ldots, k\text{-}1\}$. It first projects each instance into the real, and each ranking level has a sub-interval in the real, so the instances and ratings are combined together.

PRank tries to find a rank-prediction rule which can assign each document a ranking score which is as close as possible to the document's true rank, the basic idea is: giving an document j, the algorithm output a ranking score F(j), and F(j) will keep updating its value until the loss between the predicted rank score

and true rank score is lower than some thresholds. The algorithm detail is shown in Figure 2:

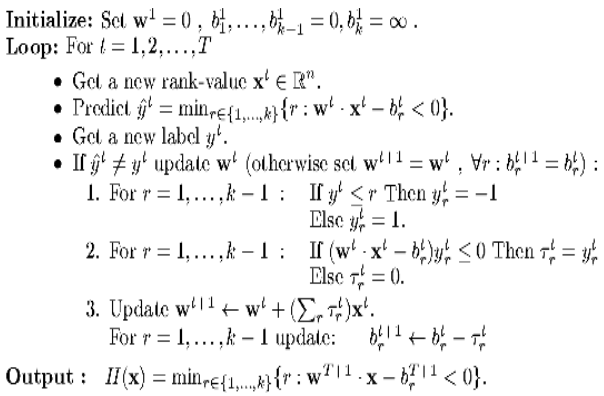

Figure 2: The PRank algorithm

Where x^t is document, y^t is the true ranking score of that document. b_i, $i \in (1,2,\ldots,k)$ is a set of thresholds. w is a vector $\in R^n$. H(x) is the ranking score of document x, documents are ranked based on this score.

4 Pairwise Approaches

In this section, we will introduce two famous pairwise methods, RankNet and LamdaRank.

3.1 RankNet

In RankNet (Chris et al., 2005), it first tries to compare the relevance of every two documents, then ranks all the documents based on the document-pair comparison result.

They defined $\overline{P_{ij}}$ to be the given probability that document i is to be ranked higher than document j, and P_{ij} to be the predicated probability that i should be ranked higher than j. They also defined $O_i = f(x_i)$, and $O_{ij} = f(x_i) - f(x_j)$, where $f(x_i)$ is the ranking score of document i. Then they used the cross entropy cost function:

$$C_{ij} = -\overline{P_{ij}} \log P_{ij} - (1 - \overline{P_{ij}}) \log(1 - P_{ij})$$

They modeled the map from outputs to probabilities using a logistic function:

$$P_{ij} = \frac{e^{O_{ij}}}{1 + e^{O_{ij}}}$$

C_{ij} then becomes:

$$C_{ij} = -\overline{P_{ij}} O_{ij} + \log(1 + e^{O_{ij}})$$

In this case, C_{ij} is plot as a function of O_{ij}, the ranking problem is casted to a optimization problem, the goal is to minimize C_{ij}. Neural nets is used to obtain O_{ij}, and two documents can be ranked based on O_{ij}.

3.2 LambdaRank

The quality measures (cost functions) used in LTR are particularly difficult to optimize directly, since they depend on the model scores only through the sorted order of the documents returned for a given query, and the cost functions are always either flat, or discontinue.

LambdaRank (Christopher et al., 2006) is a ranking algorithm based on RankNet, it tries to solve the cost function problem by working with implicit cost functions.

The main idea of LambdaRank is that, it's usually much easier to specify rules determining how we would like the rank order of documents to change, after sorting them by scores for a given query, than to construct a general, smooth optimization cost function which has the desired properties for all orderings. By only having to specify rules for a given ordering, we can define the gradients of an implicit cost function C only at the particular points in which we are interested, which is easier to find.

Based on this idea, instead of directly finding a cost function and optimizing the cost C, LambdaRank tries to find an optimization cost

C, which has the property as follow:

$$\left|\frac{\partial C}{\partial s_{j_i}}\right| \gg \left|\frac{\partial C}{\partial s_{j_k}}\right|$$

Where C is the cost function (loss function), s_{j_k} is the ranking score of document j_k, and $s_{j_k} \gg s_{j_i}$.

LambdaRank is demonstrated to be able to learn non-smooth target costs, because it doesn't need to find a perfect cost function, and it can provide a speedup for RankNet learning.

4 Listwise Approaches

In this section, we will introduce two listwise methods, ListNet and BoltzRank.

4.1 ListNet

Listnet (Zhe Cao et al., 2007) is motivated by the fact that the objective of Pairwise learning is formalized as minimizing errors in ranking document pairs, rather than minimizing errors in ranking the document list.

The idea of ListNet is similar with RankNet, instead of optimizing the pairwise loss function, it tries to optimize the listwise loss function.

They first defined a listwise loss function to represent the difference between the ranking list output by a ranking model and the ranking list given as ground truth:

$$L(y^{(i)}, z^{(i)})$$

Where $y^{(i)}$ is a list of documents' trainging ranking scores, and $z^{(i)}$ is list of documents' modeled ranking scores.

Then they introduced a top K documents probability:

$$P_s(l_k(j_1, j_2, \dots j_k)) = \prod_{t=1}^{k} \frac{\Phi(s_{j_t})}{\sum_{i=t}^{n} \Phi(s_{j_i})}$$

Where $j_1, j_2, \dots j_n$ are documents to be ranked, $l_k(j_1, j_2, \dots j_k)$ is a collection of K documents, s_{j_t} is the score of object j which is ranked in position of t, $\Phi()$ is an increasing and strictly positive function.

The listwise loss function then becomes:

$$L\left(y^{(i)}, z^{(i)}\right) = -\sum_{\forall g \in l_k} P_{y^{(i)}}(g) \log(P_{z^{(i)}}(g))$$

A new learning method for optimizing the listwise loss function based on top K priority is used, with Neural Network as model and Gradient Descent as optimization algorithm. The ranked list which has the lowest loss value will be used as the final ranked list.

4.2 BoltzRank

BoltzRank (M. N. Volkovs et al., 2009) is a new algorithm, motivated by the observation that if a probability distribution over document ranking permutations for a query can be defined, and consider the expectation of the target performance measure under this distribution, then expectation can be maximized by propagating the derivatives and update the parameters which govern the scoring function.

BoltzRank defined E(R|S):

$$E(R|S) = \frac{2}{m * (m-1)} \sum_{r_j > r_k} g_q(r_j - r_k)(s_j - s_k)$$

Where $R = \{r_1, r_2, \dots r_m\}$ is list of true scores for documents, $S = \{r_1, r_2, \dots r_m\}$ is list of scores given by the model. g_q is any sign preserving function. When $r_j > r_k$, E(R|S) gets a large negative value if $s_j < s_k$, and a

large positive one if $s_j > s_k$.

Using E(R|S), they defined the conditional Boltzmann distribution over document permutations by exponentiating and normalizing:

$$P(R|S) = \frac{1}{Z(S)}\exp(-E(R|S))$$

$$Z(S) = \sum_{R}\exp(-E(R|S))$$

The scoring function f for a document consists two potentials: individual potential Φ and pairwise potential φ.

$$f(d_j|q,D) = \Phi(d_j) + \sum_{k,k\neq j}\varphi(d_j,d_k)$$

Where d_j is the given document, q is the query and D is the document set.

Now the work is to minimize the KL divergence between the true rank distribution, P(R|L), and the model's predicated distribution P(R|S):

$$C^{R_q} = -\sum_{R\in \mathrm{R_q}} P^{R_q}(\mathrm{R|L})\log(P^{R_q}(\mathrm{R|S}))$$

The final ranking result is the ranked list which can minimize this divergence.

5 Conclusion

In this survey, we introduced three main categories of Learning to rank approaches: Pointwise Approaches, Pairwise Approaches, and Listwise Approaches. Six classic algorithms, which belong to these three categories are discussed: McRank and PRank as Pointwise Approaches, RankNet and LambdaRank as Pairwise Approaches, ListNet and BoltzRank as listwise Approaches.

The probabilistic approaches are reported to be more robust, but also more complicated. Listwise approaches are reported to have better performance, and it's easy to handle the query-specific problems. So these directions should be followed in the future work.

Other Learning to rank approaches such as graphical approaches and kernel tricks are not mentioned here, they might work well as Learning to rank methods in some specific domains.

References

C. He, C. Wang, Y. X. Zhong, and R. F. Li. (2008). A survey on Learning to Rank, In Proc. of 7th International Conference on Machine Learning and Cybernetics, July, 2008.

O. Chapelle and Y. Chang. (2011). Yahoo! Learning to Rank Challenge Overview, Journal of Machine Learning Research, vol. 14, pp. 1-24, 2011.

P. Li, C. Burges and Qiang Wu. (2007). McRank: Learning to Rank using multiple classification and gradient boosting, In Proc of NIPS 2007.

Koby Crammer and Yoram Singer. (2002). Pranking with ranking. Advances on Neural Information Processing System.

Y. Freund and R. E. Schapire. (1999). A short introduction to boosting, Journal of Japanese society for Artificial Intelligence, vol. 14, no. 5, pp. 771-780, Sept., 1999.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. (2005). Learning to rank using gradient descent. Proceedings of International Conference on Machine Learning, 2005.

T. Qin, X. D. Zhang, M. F. Tsai, D. S. Wang, T. Y. Lin and H. Li. (2008). Querylevel loss functions for Information Retrieval, Information Processing and Management, vol. 44, pp. 838-855, 2008.

Christopher Burges, Robert Ragno, and Quoc Viet Le. (2006). Learning to rank with Nonsmooth cost functions. Advances on Neural Information Processing System.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. (2007). Learning to rank: from pairwise

approach to listwise approach. Proceedings of International Conference on Machine Learning, 2007.

M. N. Volkovs and R. S. Zemel. (2009). BoltzRank: Learning to maximize expected ranking gain, In Proc. of 26th ICML, 2009.

O. Chapelle, Y. Chang and T. Y. Liu. (2011). Future directions in Learning To Rank, JMLR Workshop and Conference Proceedings, vol. 14, pp. 91-100, 2011.